// CPP program to find largest in an array

// without conditional/bitwise/ternary/ operators

// and without library functions.

#include<iostream>

#include<cstring>

#include<sstream>

using namespace std;

// If page found, updates the second chance bit to true

static bool findAndUpdate(int x,int arr[],

bool second\_chance[],int frames)

{

int i;

for(i = 0; i < frames; i++)

{

if(arr[i] == x)

{

// Mark that the page deserves a second chance

second\_chance[i] = true;

return true;

}

}

return false;

}

// Updates the page in memory and returns the pointer

static int replaceAndUpdate(int x,int arr[],

bool second\_chance[],int frames,int pointer)

{

while(true)

{

// We found the page to replace

if(!second\_chance[pointer])

{

// Replace with new page

arr[pointer] = x;

cout<<x<<" - ";

for(int i=0; i<frames;i++){

cout<<arr[i]<<" ";

}

cout<<endl;

// Return updated pointer

return (pointer + 1) % frames;

}

// Mark it 'false' as it got one chance

// and will be replaced next time unless accessed again

second\_chance[pointer] = false;

//Pointer is updated in round robin manner

pointer = (pointer + 1) % frames;

}

for(int i=0; i<frames;i++){

cout<<arr[i]<<" ";

}

cout<<endl;

}

static void printHitsAndFaults(string reference\_string,

int frames)

{

int pointer, i, l=0, x, pf;

//initially we consider frame 0 is to be replaced

pointer = 0;

//number of page faults

pf = 0;

// Create a array to hold page numbers

int arr[frames];

// No pages initially in frame,

// which is indicated by -1

memset(arr, -1, sizeof(arr));

// Create second chance array.

// Can also be a byte array for optimizing memory

bool second\_chance[frames];

// Split the string into tokens,

// that is page numbers, based on space

string str[100];

string word = "";

for (auto x : reference\_string)

{

if (x == ' ')

{

str[l]=word;

word = "";

l++;

}

else

{

word = word + x;

}

}

str[l] = word;

l++;

// l=the length of array

for(i = 0; i < l; i++)

{

x = stoi(str[i]);

// Finds if there exists a need to replace

// any page at all

if(!findAndUpdate(x,arr,second\_chance,frames))

{

// Selects and updates a victim page

pointer = replaceAndUpdate(x,arr,

second\_chance,frames,pointer);

// Update page faults

pf++;

}

}

cout << "Total page faults were " << pf << "\n";

}

// Driver code

int main()

{

string reference\_string = "";

int frames = 0;

// Test 1:

reference\_string = "0 4 1 4 2 4 3 4 2 4 0 4 1 4 2 4 3 4";

frames = 3;

// Output is 9

printHitsAndFaults(reference\_string,frames);

// Test 2:

reference\_string = "2 5 10 1 2 2 6 9 1 2 10 2 6 1 2 1 6 9 5 1";

frames = 4;

// Output is 11

printHitsAndFaults(reference\_string,frames);

return 0;

}

// This code is contributed by NikhilRathor